

ATTACHMENT A

```
1      1  top_unit
2  //-----|
3
4  module top_unit (CLK, RESET_IN, CMD_IN, CMD_OUT, CMD_OE, DATA_OE,
5  RAM_WR_STB_OUT, RAM_RD_STB_OUT, DATA_IN, HOST_RAM_DI, RAM_PTR, DATA_OUT,
6  HOST_RAM_DO, RAM_WEB, OC_PP, MP_INT, UP_DATA_IN, UP_DATA_OUT,
7  ADDR4_1, MSEL_MMCREG, RAMISB, DATA_IN_OE, CS_SELECT,
8  CS_OUT, DEBUG_MODE, SPI_CSB, MICRO_RESET, EN_SEQ_INT, SEQ_INT,
9  INT_REG_ADDR, TOGGLEHOSTBUF, RESET_TO_HC11
10 , HC11_CLK, MWR_MMC_LDS, MICRO_RWB);
11
12 //to _make DIR_OUT what to do with it in thd "cmd_unit" components
13 //to _make last bit in the cid in the memory is the end bit
14
15 `include "param.v"
16 input CLK;
17 input RESET_IN; //async. reset
18 input DATA_IN; //data line in
19 input CMD_IN; //cmd line in
20 input [15:0]HOST_RAM_DO; //data from host to data shift register
21 input HC11_CLK; // Address Strobe from CPU
22 input [7:0]UP_DATA_OUT;
23 input MWR_MMC_LDS;
24 input MICRO_RWB;
25 input [4:1]ADDR4_1;
26 input MSEL_MMCREG;
27
28 input [2:0]CS_SELECT;
29 input DEBUG_MODE;
30 input SPI_CSB;
31 input EN_SEQ_INT;
32 input SEQ_INT;
33 input TOGGLEHOSTBUF; //toggle from seq
34 input RESET_TO_HC11;
35
36 output RAM_WR_STB_OUT; //strobe signal to host ram
37 output RAM_RD_STB_OUT; //strobe signal to host ram
38 output [15:0]HOST_RAM_DI; //data shift reg to DI of host ram
39 output [8:0]RAM_PTR; //address to host ram
40 output DATA_OUT; //data out from data shift register
41 output CMD_OUT;
42 output CMD_OE;
43 output DATA_OE;
44 output RAM_WEB;
45 output OC_PP;
46 output MP_INT;
47 output [7:0]UP_DATA_IN;
48 output RAMISB;
49
50 output DATA_IN_OE;
51 output MICRO_RESET;
52 output INT_REG_ADDR;
53 output [5:0]CS_OUT;
54
```

```

55     wire SPI_CYCLE_5;
56     wire SPI_CYCLE_6;
57 //top_lgc
58 //output
59     wire RESET; //sync. reset
60
61
62 //reg_unit
63 //output
64     wire [7:0]BE_ERROR_BUS;
65     wire BE_ERROR;
66     wire BE_DONE;
67     wire XMT_BUF_FULL;
68     wire RCV_BUF_EMPTY;
69     wire ANY_BLK_LEN;
70     wire STRM;
71     wire TEST_MODE;
72
73
74 //cmd_unit
75 //output
76
77 //cmd_unit
78     wire CMD;
79     wire [9:0]BLOCKLEN_REG;
80     wire [5:0]BE_CMD_REG;
81     wire RCA_MATCH;
82     wire [15:0]SECTOR_ADDR_REG;
83     wire BLOCKLEN_OK;
84     wire [39:0]CMD_BUS;
85     wire PRG_CMD;
86     wire TAG_CMD;
87     wire FLASH_WR_CMD;
88     wire DATA_CMD;
89     wire STOP_TRANSMISSION;
90     wire RESPONSE_DONE_FROM_CMD_UNIT;
91     wire START_ADDRESS_OK;
92     wire GO_IDLE;
93     wire GLOBAL_CMD;
94     wire COMPATIBLE;
95     wire OCR_IN_IS_0;
96     wire [8:0]ARG_8_TO_0;
97     wire CID_ERROR;
98     wire RESPONSE_START;
99     wire CMD_OUT_TO_BUS_CONT_UNIT;
100    wire [10:0]LOCKED_CMD_BUS;
101    wire SET_CMD_DIR_OUT_FROM_CMD_UNIT;
102    wire LD_END_ON_DATA;
103    wire HOLD_XMT_FSM;
104    wire CLR_DATA_OE_FROM_CMD_UNIT;
105    wire SET_DATA_OE_FROM_CMD_UNIT;
106    wire BUSY_FLAG;
107    wire ALL_CLR_BUSY;
108    wire CLR_MAIN_OE;
109    wire SET_MAIN_OE;
110    wire CLR_OC_PP;

```

```

111 wire SET_OC_PP;
112 wire BWR_IS_1;
113 wire GO_TEST;
114 wire TEST_CMD;
115 wire CHECK_FOR_CRC;
116 wire CMD_CRC_MSB;
117 wire CMD_SR_MSB;
118 wire BLOCKLEN_OK_FOR_WRITE;
119 wire [15:0]ARG_24_TO_9;
120 wire [5:0]CS_FROM_CMD_CTL;
121
122 //output
123 //cont_unit
124 wire SET_MP_INT;
125 wire SET_OCR_BUSY;
126 wire SEND_OCR_RESPONSE;
127 wire SEND_RESPONSE;
128 wire CLR_OCR_BUSY;
129 wire LD_BE_ERROR_REG;
130 wire CID_MODE;
131 wire LD_RCA_REG;
132 wire [3:0]CURRENT_STATE;
133 wire LD_DSR_REG;
134 wire SEND_CID_RESPONSE;
135 wire SET_ERASE_RESET;
136 wire LD_BLOCKLEN_REG;
137 wire SET_BLOCK_LEN_ERROR;
138 wire LD_0_PNG_PTR;
139 wire INC_PNG_PTR;
140 wire SET_ERASE_SEQ_ERROR;
141 wire RAM_WR_STB_FROM_CONT_UNIT;
142 wire LD_ADDR_STATUS_BITS;
143 wire CLR_BE_DONE;
144 wire CLR_BE_ERROR;
145 wire CLR_STOP_MP_RD;
146 wire STOP_XMT_UNIT;
147 wire SET_STOP_MP_RD;
148 wire CLR_STOP_TRANSMISSION_FROM_CONT_UNIT;
149 wire GAP_XMT_UNIT;
150 wire SET_XMT_BUF_EMPTY;
151 wire CLR_XMT_BUF_FULL;
152 wire START_XMT_UNIT;
153 wire SET_UNDERRUN_ERROR;
154 wire SET_ADDRESS_ERROR_FROM_CONT_UNIT;
155 wire WP_MODE;
156 wire STOP_RCV_UNIT;
157 wire CLR_LAST_BUF;
158 wire CLR_RCV_BUF_FULL;
159 wire CLR_RCV_BUF_EMPTY;
160 wire START_RCV_UNIT;
161 wire SET_OVERRUN_ERROR;
162 wire SET_RCV_BUF_FULL;
163 wire SET_LAST_BUF;
164 wire CLR_MISALIGN_ERROR;
165 wire SELECT_CARD_IF_RCA_MATCH;
166 wire SELECT_CARD_IF_NOT_RCA_MATCH;

```

```

167 wire SET_DATA_OE_FROM_CONT_UNIT;
168 wire CLR_DATA_OE_FROM_CONT_UNIT;
169 wire LD_GAP_FROM_CONT_UNIT;
170 wire SEND_RESPONSE_FROM_CONT_UNIT;
171 wire LOCK_CMD_BUS;
172 wire CLR_BUSY_FROM_CONT_UNIT;
173 wire CLR_WRITE_CYCLE;
174 wire LD_ADDR_RANGE;
175 wire CLR_ADDR_RANGE;
176 wire TAG_SECTOR;
177 wire SET_BWR_ERROR;
178 wire TOGGLE_BUF;
179 wire SET_BUF;
180 wire CLR_BUF;
181 wire SET_RAM_WEB_FROM_CONT_UNIT;
182 wire CLR_RAM_WEB_FROM_CONT_UNIT;
183 wire SET_CID_OVERWRITE;
184 wire STOP_WR_FSM;
185 wire ALL_FSM_INACTIVE;
186 wire SEND_RESPONSE_SPI_BE;
187 wire LD_CHECK_CRC_REG;
188 wire SPI_DATA_CYCLE;
189 wire CLR_MICRO_RESET;
190 wire [4:0]CS_FROM_CONT_FSM_A;
191 wire [4:0]CS_FROM_CONT_FSM_B;
192 wire [5:0]CS_FROM_CONT_FSM_C;
193 wire [3:0]CS_FROM_CONT_WR_FSM;
194 wire [4:0]CS_FROM_CONT_RD_FSM;
195
196
197
198 //data_unit
199 //output
200 wire CLR_STOP_TRANSMISSION_FROM_DATA_UNIT;
201 wire RCV_UNIT_DONE;
202 wire SET_ADDRESS_ERROR_FROM_DATA_UNIT;
203 wire XMT_UNIT_DONE;
204 wire CID_END;
205 wire RD_MISALIGN_ERROR;
206 wire CID_DATA;
207 wire CLR_CID_CHECK;
208 wire SET_CID_CHECK;
209 wire OVER_100_UNTAG;
210 wire [9:0]RCV_UNIT_BUF_START_ADDR;
211 wire [9:0]RCV_UNIT_BUF_END_ADDR;
212 wire RESPONSE_DONE_FROM_DATA_UNIT;
213 wire CLR_DATA_OE_FROM_DATA_UNIT;
214 wire DATA_OUT_TO_CMD_UNIT;
215 wire SET_DATA_OE_FROM_DATA_UNIT;
216 wire SET_FINAL_RCV_UNIT_DONE;
217 wire SET_BUSY_FROM_DATA_UNIT;
218 wire CLR_BUSY_FROM_DATA_UNIT;
219 wire DATA_CRC_MSB;
220 wire DATA_SR_MSB;
221 wire [9:0]DYNAMIC_RCV_UNIT_BUF_START_ADDR;
222 wire [9:0]DYNAMIC_RCV_UNIT_BUF_END_ADDR;

```

```

223 wire [5:0]CS_FROM_XMT_FSM;
224 wire [5:0]CS_FROM_RCV_FSM;
225
226 //cont_bus_unit
227 //output
228 wire OUT_CYCLE_IS_DATA;
229 wire SPI_DATA_LINE_ACTIVE;
230
231 //top_lgc
232 //output
233 wire CMD_IN_W_SPI;
234 wire CMD_OUT_WO_SPI;
235 wire DATA_IN_W_SPI;
236 wire DATA_OUT_WO_SPI;
237 wire PAR_RESET;
238
239
240
241 top_lgc u_top_lgc(.CLK(CLK), .RESET_IN(RESET_IN), .GO_IDLE(GO_IDLE),
242 .RESET(RESET), .SPI_MODE(SPI_MODE), .SPI_CSB(SPI_CSB),
243 .SPI_CYCLE_5(SPI_CYCLE_5), .SPI_CYCLE_6(SPI_CYCLE_6),
244 .CMD_IN(CMD_IN), .CMD_OUT_WO_SPI(CMD_OUT_WO_SPI), .DATA_IN(DATA_IN),
245 .DATA_OUT(DATA_OUT), .SR_3(SR_3),
246 .CMD_IN_W_SPI(CMD_IN_W_SPI), .CMD_OUT(CMD_OUT),
247 .DATA_IN_W_SPI(DATA_IN_W_SPI), .DATA_OUT_WO_SPI(DATA_OUT_WO_SPI),
248 .CMD_OE(CMD_OE), .DATA_OE(DATA_OE),
249 .SPI_DATA_CYCLE(SPI_DATA_CYCLE),
250 .CS_FROM_CONT_FSM_A(CS_FROM_CONT_FSM_A),
251 .CS_FROM_CONT_FSM_B(CS_FROM_CONT_FSM_B),
252 .CS_FROM_CONT_FSM_C(CS_FROM_CONT_FSM_C),
253 .CS_FROM_CONT_WR_FSM(CS_FROM_CONT_WR_FSM),
254 .CS_FROM_CONT_RD_FSM(CS_FROM_CONT_RD_FSM),
255 .CS_FROM_XMT_FSM(CS_FROM_XMT_FSM),
256 .CS_FROM_RCV_FSM(CS_FROM_RCV_FSM),
257 .CS_FROM_CMD_CTL(CS_FROM_CMD_CTL), .CS_SELECT(CS_SELECT), .CS_OUT(CS_OUT),
258 .DEBUG_MODE(DEBUG_MODE), .DATA_CRC_MSB(DATA_CRC_MSB),
259 .CMD_CRC_MSB(CMD_CRC_MSB), .DATA_SR_MSB(DATA_SR_MSB),
260 .CMD_SR_MSB(CMD_SR_MSB), .OUT_CYCLE_IS_DATA(OUT_CYCLE_IS_DATA),
261 .SPI_DATA_LINE_ACTIVE(SPI_DATA_LINE_ACTIVE), .PAR_RESET(PAR_RESET));
262
263 cmd_unit u_cmd_unit(.CLK(CLK), .RESET(RESET), .CMD(CMD), .CMD_IN(CMD_IN_W_SPI),
264 .CMD_OUT(CMD_OUT_TO_BUS_CONT_UNIT), .BE_ERROR_BUS(BE_ERROR_BUS),
265 .DATA_OUT(DATA_OUT_TO_CMD_UNIT), .CID_DATA(CID_DATA),
266 .LD_BLOCKLEN_REG(LD_BLOCKLEN_REG), .LD_RCA_REG(LD_RCA_REG),
267 .LD_DSR_REG(LD_DSR_REG), .SET_MP_INT(SET_MP_INT),
268 .CLR_STOP_TRANSMISSION_FROM_CONT_UNIT(CLR_STOP_TRANSMISSION_FROM_CONT_UNIT),
269 .CLR_STOP_TRANSMISSION_FROM_DATA_UNIT(CLR_STOP_TRANSMISSION_FROM_DATA_UNIT),
270 .CID_MODE(CID_MODE), .WP_MODE(WP_MODE), .SEND_CID_RESPONSE(SEND_CID_RESPONSE),
271 .SEND_OCR_RESPONSE(SEND_OCR_RESPONSE), .SEND_RESPONSE(SEND_RESPONSE),
272 .BLOCKLEN_REG(BLOCKLEN_REG), .BE_CMD_REG(BE_CMD_REG), .RCA_MATCH(RCA_MATCH),
273 .SECTOR_ADDR_REG(SECTOR_ADDR_REG),
274 .BLOCKLEN_OK(BLOCKLEN_OK), .CMD_BUS(CMD_BUS), .PRG_CMD(PRG_CMD),
275 .TAG_CMD(TAG_CMD), .FLASH_WR_CMD(FLASH_WR_CMD),
276 .DATA_CMD(DATA_CMD), .STOP_TRANSMISSION(STOP_TRANSMISSION),
277 .RESPONSE_DONE(RESPONSE_DONE_FROM_CMD_UNIT),
278 .START_ADDRESS_OK(START_ADDRESS_OK), .GO_IDLE(GO_IDLE), .GLOBAL_CMD(GLOBAL_CMD),

```

```

279 .SET_OCR_BUSY(SET_OCR_BUSY), .CLR_OCR_BUSY(CLR_OCR_BUSY),
280 .SET_BLOCK_LEN_ERROR(SET_BLOCK_LEN_ERROR), .SET_ERASE_RESET(SET_ERASE_RESET),
281 .SET_ERASE_SEQ_ERROR(SET_ERASE_SEQ_ERROR),
282 .LD_ADDR_STATUS_BITS(LD_ADDR_STATUS_BITS), .LD_BE_ERROR_REG(LD_BE_ERROR_REG),
283 .SET_ADDRESS_ERROR_FROM_DATA_UNIT(SET_ADDRESS_ERROR_FROM_DATA_UNIT),
284 .SET_ADDRESS_ERROR_FROM_CONT_UNIT(SET_ADDRESS_ERROR_FROM_CONT_UNIT),
285 .SET_UNDERRUN_ERROR(SET_UNDERRUN_ERROR),
286 .SET_OVERRUN_ERROR(SET_OVERRUN_ERROR),
287 .CURRENT_STATE(CURRENT_STATE),
288 .COMPATIBLE(COMPATIBLE), .OCR_IN_IS_0(OCR_IN_IS_0), .ARG_8_TO_0(ARG_8_TO_0),
289 .CLR_CID_CHECK(CLR_CID_CHECK), .SET_CID_CHECK(SET_CID_CHECK),
290 .CID_ERROR(CID_ERROR), .SELECT_CARD_IF_RCA_MATCH(SELECT_CARD_IF_RCA_MATCH),
291 .SELECT_CARD_IF_NOT_RCA_MATCH(SELECT_CARD_IF_NOT_RCA_MATCH),
292 .RESPONSE_START(RESPONSE_START),
293 .RESPONSE_DONE_FROM_DATA_UNIT(RESPONSE_DONE_FROM_DATA_UNIT),
294 .SEND_RESPONSE_FROM_CONT_UNIT(SEND_RESPONSE_FROM_CONT_UNIT),
295 .LOCK_CMD_BUS(LOCK_CMD_BUS),
296 .LOCKED_CMD_BUS(LOCKED_CMD_BUS),
297 .SET_CMD_DIR_OUT(SET_CMD_DIR_OUT_FROM_CMD_UNIT),
298 .LD_END_ON_DATA(LD_END_ON_DATA), .HOLD_XMT_FSM(HOLD_XMT_FSM),
299 .DATA_OE(DATA_OE), .CLR_DATA_OE(CLR_DATA_OE_FROM_CMD_UNIT),
300 .SET_DATA_OE(SET_DATA_OE_FROM_CMD_UNIT), .SR_3(SR_3), .SPI_CSB(SPI_CSB),
301 .SET_BUSY_FROM_DATA_UNIT(SET_BUSY_FROM_DATA_UNIT),
302 .CLR_BUSY_FROM_DATA_UNIT(CLR_BUSY_FROM_DATA_UNIT),
303 .CLR_BUSY_FROM_CONT_UNIT(CLR_BUSY_FROM_CONT_UNIT), .BUSY_FLAG(BUSY_FLAG),
304 .ALL_CLR_BUSY(ALL_CLR_BUSY), .READY_TO_ERASE(READY_TO_ERASE),
305 .CLR_MAIN_OE(CLR_MAIN_OE), .SET_MAIN_OE(SET_MAIN_OE), .CLR_OC_PP(CLR_OC_PP),
306 .SET_OC_PP(SET_OC_PP), .BWR_IS_1(BWR_IS_1), .SET_BWR_ERROR(SET_BWR_ERROR),
307 .STRM(STRM), .ANY_BLK_LEN(ANY_BLK_LEN), .GO_TEST(GO_TEST),
308 .TEST_MODE(TEST_MODE), .TEST_CMD(TEST_CMD),
309 .SET_CID_OVERWRITE(SET_CID_OVERWRITE), .ARG_24_TO_9(ARG_24_TO_9),
310 .SPI_MODE(SPI_MODE), .SPI_CYCLE_6(SPI_CYCLE_6),
311 .SEND_RESPONSE_SPI_BE(SEND_RESPONSE_SPI_BE),
312 .LD_CHECK_CRC_REG(LD_CHECK_CRC_REG), .CHECK_FOR_CRC(CHECK_FOR_CRC),
313 .SPI_DATA_CYCLE(SPI_DATA_CYCLE),
314 .CS_FROM_CMD_CTL(CS_FROM_CMD_CTL), .CMD_CRC_MSB(CMD_CRC_MSB),
315 .CMD_SR_MSB(CMD_SR_MSB), .BLOCKLEN_OK_FOR_WRITE(BLOCKLEN_OK_FOR_WRITE),
316 .PAR_RESET(PAR_RESET));
317
318
319 cont_unit u_cont_unit(.CLK(CLK), .RESET(RESET), .GLOBAL_CMD(GLOBAL_CMD),
320 .RCA_MATCH(RCA_MATCH),
321 .CMD_BUS(CMD_BUS), .CMD(CMD), .SET_MP_INT(SET_MP_INT),
322 .GO_IDLE(GO_IDLE),
323 .COMPATIBLE(COMPATIBLE), .OCR_IN_IS_0(OCR_IN_IS_0),
324 .SET_OCR_BUSY(SET_OCR_BUSY),
325 .SEND_OCR_RESPONSE(SEND_OCR_RESPONSE), .CLR_OCR_BUSY(CLR_OCR_BUSY),
326 .LD_BE_ERROR_REG(LD_BE_ERROR_REG), .SEND_RESPONSE(SEND_RESPONSE),
327 .CID_MODE(CID_MODE),
328 .CID_END(CID_END), .CID_ERROR(CID_ERROR), .LD_RCA_REG(LD_RCA_REG),
329 .CURRENT_STATE(CURRENT_STATE), .TAG_CMD(TAG_CMD),
330 .BLOCKLEN_OK(BLOCKLEN_OK), .DATA_CMD(DATA_CMD), .PRG_CMD(PRG_CMD),
331 .LD_DSR_REG(LD_DSR_REG), .SEND_CID_RESPONSE(SEND_CID_RESPONSE),
332 .SET_ERASE_RESET(SET_ERASE_RESET), .LD_BLOCKLEN_REG(LD_BLOCKLEN_REG),
333 .SET_BLOCK_LEN_ERROR(SET_BLOCK_LEN_ERROR), .LD_0_PNG_PTR(LD_0_PNG_PTR),
334 .INC_PNG_PTR(INC_PNG_PTR), .SET_ERASE_SEQ_ERROR(SET_ERASE_SEQ_ERROR),

```

```

335 .RAM_WR_STB(RAM_WR_STB_FROM_CONT_UNIT), .OVER_100_UNTAG(OVER_100_UNTAG),
336 .START_ADDRESS_OK(START_ADDRESS_OK), .BE_ERROR(BE_ERROR),
337 .STOP_TRANSMISSION(STOP_TRANSMISSION),
338 .BE_DONE(BE_DONE), .LD_ADDR_STATUS_BITS(LD_ADDR_STATUS_BITS),
339 .CLR_BE_DONE(CLR_BE_DONE), .CLR_BE_ERROR(CLR_BE_ERROR),
340 .CLR_STOP_MP_RD(CLR_STOP_MP_RD), .STOP_XMT_UNIT(STOP_XMT_UNIT),
341 .SET_STOP_MP_RD(SET_STOP_MP_RD),
342 .CLR_STOP_TRANSMISSION(CLR_STOP_TRANSMISSION_FROM_CONT_UNIT),
343 .GAP_XMT_UNIT(GAP_XMT_UNIT),
344 .XMT_BUF_FULL(XMT_BUF_FULL), .XMT_UNIT_DONE(XMT_UNIT_DONE),
345 .MISALIGN_ERROR(RD_MISALIGN_ERROR), .SET_XMT_BUF_EMPTY(SET_XMT_BUF_EMPTY),
346 .CLR_XMT_BUF_FULL(CLR_XMT_BUF_FULL), .TOGGLE_BUF(TOGGLE_BUF),
347 .START_XMT_UNIT(START_XMT_UNIT), .SET_BUF(SET_BUF), .CLR_BUF(CLR_BUF),
348 .SET_UNDERRUN_ERROR(SET_UNDERRUN_ERROR),
349 .SET_ADDRESS_ERROR(SET_ADDRESS_ERROR_FROM_CONT_UNIT), .WP_MODE(WP_MODE),
350 .STOP_RCV_UNIT(STOP_RCV_UNIT), .CLR_LAST_BUF(CLR_LAST_BUF),
351 .CLR_RCV_BUF_FULL(CLR_RCV_BUF_FULL), .CLR_RCV_BUF_EMPTY(CLR_RCV_BUF_EMPTY),
352 .START_RCV_UNIT(START_RCV_UNIT), .SET_OVERRUN_ERROR(SET_OVERRUN_ERROR),
353 .SET_RCV_BUF_FULL(SET_RCV_BUF_FULL), .SET_LAST_BUF(SET_LAST_BUF),
354 .RCV_UNIT_DONE(RCV_UNIT_DONE), .RCV_BUF_EMPTY(RCV_BUF_EMPTY),
355 .CLR_MISALIGN_ERROR(CLR_MISALIGN_ERROR),
356 .SELECT_CARD_IF_RCA_MATCH(SELECT_CARD_IF_RCA_MATCH),
357 .SELECT_CARD_IF_NOT_RCA_MATCH(SELECT_CARD_IF_NOT_RCA_MATCH),
358 .SET_DATA_OE(SET_DATA_OE_FROM_CONT_UNIT),
359 .CLR_DATA_OE(CLR_DATA_OE_FROM_CONT_UNIT), .LD_GAP(LD_GAP_FROM_CONT_UNIT),
360 .SEND_RESPONSE_FROM_CONT_UNIT(SEND_RESPONSE_FROM_CONT_UNIT),
361 .LOCK_CMD_BUS(LOCK_CMD_BUS),
362 .LOCKED_CMD_BUS(LOCKED_CMD_BUS),
363 .SET_FINAL_RCV_UNIT_DONE(SET_FINAL_RCV_UNIT_DONE),
364 .CLR_BUSY(CLR_BUSY_FROM_CONT_UNIT), .CLR_WRITE_CYCLE(CLR_WRITE_CYCLE),
365 .RESPONSE_DONE_FROM_CMD_UNIT(RESPONSE_DONE_FROM_CMD_UNIT),
366 .LD_ADDR_RANGE(LD_ADDR_RANGE), .CLR_ADDR_RANGE(CLR_ADDR_RANGE),
367 .READY_TO_ERASE(READY_TO_ERASE), .TAG_SECTOR(TAG_SECTOR), .BWR_IS_1(BWR_IS_1),
368 .SET_BWR_ERROR(SET_BWR_ERROR), .GO_TEST(GO_TEST),
369 .SET_RAM_WEB(SET_RAM_WEB_FROM_CONT_UNIT),
370 .CLR_RAM_WEB(CLR_RAM_WEB_FROM_CONT_UNIT), .TEST_MODE(TEST_MODE),
371 .SET_CID_OVERWRITE(SET_CID_OVERWRITE), .STOP_WR_FSM(STOP_WR_FSM),
372 .SPI_MODE(SPI_MODE), .SPI_CYCLE_5(SPI_CYCLE_5),
373 .ALL_FSM_INACTIVE(ALL_FSM_INACTIVE),
374 .SEND_RESPONSE_SPI_BE(SEND_RESPONSE_SPI_BE),
375 .LD_CHECK_CRC_REG(LD_CHECK_CRC_REG), .SPI_DATA_CYCLE(SPI_DATA_CYCLE),
376 .CS_FROM_CONT_FSM_A(CS_FROM_CONT_FSM_A),
377 .CS_FROM_CONT_FSM_B(CS_FROM_CONT_FSM_B),
378 .CS_FROM_CONT_FSM_C(CS_FROM_CONT_FSM_C),
379 .CS_FROM_CONT_WR_FSM(CS_FROM_CONT_WR_FSM),
380 .CS_FROM_CONT_RD_FSM(CS_FROM_CONT_RD_FSM),
381 .BLOCKLEN_OK_FOR_WRITE(BLOCKLEN_OK_FOR_WRITE),
382 .CLR_MICRO_RESET(CLR_MICRO_RESET));
383
384
385 data_unit u_data_unit(.CLK(CLK), .RESET(RESET), .STOP_RCV_UNIT(STOP_RCV_UNIT),
386 .START_RCV_UNIT(START_RCV_UNIT), .LOCKED_CMD_BUS(LOCKED_CMD_BUS),
387 .STOP_TRANSMISSION(STOP_TRANSMISSION),
388 .CLR_STOP_TRANSMISSION(CLR_STOP_TRANSMISSION_FROM_DATA_UNIT),
389 .RCV_UNIT_DONE(RCV_UNIT_DONE), .RAM_WR_STB_OUT(RAM_WR_STB_OUT),
390 .RAM_RD_STB_OUT(RAM_RD_STB_OUT),

```

```

391 .SET_ADDRESS_ERROR(SET_ADDRESS_ERROR_FROM_DATA_UNIT), .DATA_IN(DATA_IN_W_SPI),
392 .HOST_RAM_DI(HOST_RAM_DI), .BLOCKLEN_REG(BLOCKLEN_REG),
393 .ARG_8_TO_0(ARG_8_TO_0), .RAM_PTR(RAM_PTR),
394 .RCV_UNIT_BUF_START_ADDR(RCV_UNIT_BUF_START_ADDR),
395 .RCV_UNIT_BUF_END_ADDR(RCV_UNIT_BUF_END_ADDR), .GAP_XMT_UNIT(GAP_XMT_UNIT),
396 .START_XMT_UNIT(START_XMT_UNIT), .XMT_UNIT_DONE(XMT_UNIT_DONE),
397 .CID_END(CID_END), .RAM_WR_STB_FROM_CONT_UNIT(RAM_WR_STB_FROM_CONT_UNIT),
398 .CID_MODE(CID_MODE), .WP_MODE(WP_MODE), .RD_MISALIGN_ERROR(RD_MISALIGN_ERROR),
399 .CLR_MISALIGN_ERROR(CLR_MISALIGN_ERROR), .CID_DATA(CID_DATA),
400 .DATA_OUT(DATA_OUT_WO_SPI), .HOST_RAM_DO(HOST_RAM_DO),
401 .CLR_CID_CHECK(CLR_CID_CHECK),
402 .SET_CID_CHECK(SET_CID_CHECK), .CID_ERROR(CID_ERROR),
403 .OVER_100_UNTAG(OVER_100_UNTAG), .LD_0_PNG_PTR(LD_0_PNG_PTR),
404 .INC_PNG_PTR(INC_PNG_PTR), .STOP_XMT_UNIT(STOP_XMT_UNIT),
405 .RESPONSE_DONE(RESPONSE_DONE_FROM_DATA_UNIT),
406 .CLR_DATA_OE(CLR_DATA_OE_FROM_DATA_UNIT),
407 .LD_GAP_FROM_CONT_UNIT(LD_GAP_FROM_CONT_UNIT),
408 .DATA_OUT_TO_CMD_UNIT(DATA_OUT_TO_CMD_UNIT), .LD_END_ON_DATA(LD_END_ON_DATA),
409 .HOLD_XMT_FSM(HOLD_XMT_FSM), .SET_DATA_OE(SET_DATA_OE_FROM_DATA_UNIT),
410 .SET_FINAL_RCV_UNIT_DONE(SET_FINAL_RCV_UNIT_DONE),
411 .SET_BUSY(SET_BUSY_FROM_DATA_UNIT),
412 .CLR_BUSY(CLR_BUSY_FROM_DATA_UNIT), .CLR_WRITE_CYCLE(CLR_WRITE_CYCLE),
413 .BUSY_FLAG(BUSY_FLAG), .ALL_CLR_BUSY(ALL_CLR_BUSY),
414 .LD_ADDR_RANGE(LD_ADDR_RANGE), .RAM_WEB(RAM_WEB),
415 .CLR_ADDR_RANGE(CLR_ADDR_RANGE), .CMD_BUS(CMD_BUS), .TAG_CMD(TAG_CMD),
416 .TEST_CMD(TEST_CMD), .SET_RAM_WEB_FROM_CONT_UNIT(SET_RAM_WEB_FROM_CONT_UNIT),
417 .CLR_RAM_WEB_FROM_CONT_UNIT(CLR_RAM_WEB_FROM_CONT_UNIT),
418 .STOP_WR_FSM(STOP_WR_FSM), .ARG_24_TO_9(ARG_24_TO_9),
419 .DYNAMIC_RCV_UNIT_BUF_START_ADDR(DYNAMIC_RCV_UNIT_BUF_START_ADDR),
420 .DYNAMIC_RCV_UNIT_BUF_END_ADDR(DYNAMIC_RCV_UNIT_BUF_END_ADDR),
421 .SET_LAST_BUF(SET_LAST_BUF), .SPI_CYCLE_5(SPI_CYCLE_5),
422 .CHECK_FOR_CRC(CHECK_FOR_CRC), .CS_FROM_XMT_FSM(CS_FROM_XMT_FSM),
423 .CS_FROM_RCV_FSM(CS_FROM_RCV_FSM), .DATA_CRC_MSB(DATA_CRC_MSB),
424 .DATA_SR_MSB(DATA_SR_MSB), .SPI_MODE(SPI_MODE));
425
426
427 reg_unit u_reg_unit(.CLK(CLK), .PAR_RESET(PAR_RESET), .RESET(RESET),
428 .CLR_BE_DONE(CLR_BE_DONE),
429 .CLR_BE_ERROR(CLR_BE_ERROR), .CLR_STOP_MP_RD(CLR_STOP_MP_RD),
430 .SET_STOP_MP_RD(SET_STOP_MP_RD), .SET_XMT_BUF_EMPTY(SET_XMT_BUF_EMPTY),
431 .CLR_XMT_BUF_FULL(CLR_XMT_BUF_FULL), .CLR_LAST_BUF(CLR_LAST_BUF),
432 .CLR_RCV_BUF_FULL(CLR_RCV_BUF_FULL), .CLR_RCV_BUF_EMPTY(CLR_RCV_BUF_EMPTY),
433 .SET_RCV_BUF_FULL(SET_RCV_BUF_FULL), .SET_LAST_BUF(SET_LAST_BUF),
434 .BE_ERROR_BUS(BE_ERROR_BUS), .BE_ERROR(BE_ERROR), .BE_DONE(BE_DONE),
435 .XMT_BUF_FULL(XMT_BUF_FULL), .RCV_BUF_EMPTY(RCV_BUF_EMPTY),
436 .RCV_UNIT_BUF_START_ADDR(RCV_UNIT_BUF_START_ADDR),
437 .RCV_UNIT_BUF_END_ADDR(RCV_UNIT_BUF_END_ADDR), .SET_MP_INT(SET_MP_INT),
438 .BE_CMD_REG(BE_CMD_REG), .SECTOR_ADDR_REG(SECTOR_ADDR_REG),
439 .TAG_SECTOR(TAG_SECTOR), .ANY_BLK_LEN(ANY_BLK_LEN), .MP_INT(MP_INT),
440 .UP_DATA_IN(UP_DATA_IN), .UP_DATA_OUT(UP_DATA_OUT),
441 .ADDR4_1(ADDR4_1), .MSEL_MMCREG(MSEL_MMCREG), .RAMISB(RAMISB),
442 .TOGGLE_BUF(TOGGLE_BUF),
443 .SET_BUF(SET_BUF), .CLR_BUF(CLR_BUF), .STRM(STRM), .DATA_IN_OE(DATA_IN_OE),
444 .TEST_MODE(TEST_MODE),
445 .DYNAMIC_RCV_UNIT_BUF_START_ADDR(DYNAMIC_RCV_UNIT_BUF_START_ADDR),
446 .DYNAMIC_RCV_UNIT_BUF_END_ADDR(DYNAMIC_RCV_UNIT_BUF_END_ADDR),

```



```

447 .MICRO_RESET(MICRO_RESET), .CLR_MICRO_RESET(CLR_MICRO_RESET),
448 .EN_SEQ_INT(EN_SEQ_INT), .SEQ_INT(SEQ_INT),
449 .INT_REG_ADDR(INT_REG_ADDR),
450 .TOGGLEHOSTBUF(TOGGLEHOSTBUF),
451 .HC11_CLK(HC11_CLK), .MWR MMC_LDS(MWR MMC_LDS), .MICRO_RWB(MICRO_RWB),
452 .RESET_TO_HC11(RESET_TO_HC11));
453
454
455
456
457 bus_cont_unit u_bus_cont_unit(.CLK(CLK), .RESET(RESET),
458 .SET_CMD_DIR_OUT_FROM_CMD_UNIT(SET_CMD_DIR_OUT_FROM_CMD_UNIT),
459 .SET_CMD_DIR_OUT(SEND_RESPONSE_FROM_CONT_UNIT),
460 .SET_CMD_DIR_IN_FROM_CMD_UNIT(RESPONSE_DONE_FROM_CMD_UNIT),
461 .SET_CMD_DIR_IN_FROM_DATA_UNIT(RESPONSE_DONE_FROM_DATA_UNIT), .CMD_OE(CMD_OE),
462 .CMD_OUT_TO_BUS_CONT_UNIT(CMD_OUT_TO_BUS_CONT_UNIT), .CMD_OUT(CMD_OUT_WO_SPI),
463 .CID_ERROR(CID_ERROR), .SET_DATA_OE_FROM_CONT_UNIT(SET_DATA_OE_FROM_CONT_UNIT),
464 .CLR_DATA_OE_FROM_CONT_UNIT(CLR_DATA_OE_FROM_CONT_UNIT),
465 .CLR_DATA_OE_FROM_DATA_UNIT(CLR_DATA_OE_FROM_DATA_UNIT), .DATA_OE(DATA_OE),
466 .CLR_DATA_OE_FROM_CMD_UNIT(CLR_DATA_OE_FROM_CMD_UNIT),
467 .SET_DATA_OE_FROM_CMD_UNIT(SET_DATA_OE_FROM_CMD_UNIT),
468 .SET_DATA_OE_FROM_DATA_UNIT(SET_DATA_OE_FROM_DATA_UNIT),
469 .CLR_MAIN_OE(CLR_MAIN_OE), .SET_MAIN_OE(SET_MAIN_OE), .CLR_OC_PP(CLR_OC_PP),
470 .SET_OC_PP(SET_OC_PP), .OC_PP(OC_PP), .ALL_FSM_INACTIVE(ALL_FSM_INACTIVE),
471 .SPI_MODE(SPI_MODE), .SPI_CSB(SPI_CSB),
472 .OUT_CYCLE_IS_DATA(OUT_CYCLE_IS_DATA), .LOCKED_CMD_BUS(LOCKED_CMD_BUS),
473 .SPI_DATA_LINE_ACTIVE(SPI_DATA_LINE_ACTIVE), .PAR_RESET(PAR_RESET));
474
475 endmodule // top_unit

```

```

476         2    top_lgc
477 //-----|
478 module top_lgc (CLK, RESET_IN, GO_IDLE, RESET, SPI_MODE, SPI_CSB, SPI_CYCLE_5,
479                SPI_CYCLE_6, CMD_IN, CMD_OUT, DATA_IN, DATA_OUT,
480                CMD_IN_W_SPI, CMD_OUT_WO_SPI, DATA_IN_W_SPI, DATA_OUT_WO_SPI,
481                CMD_OE, DATA_OE, SPI_DATA_CYCLE,
482                CS_FROM_CONT_FSM_A, CS_FROM_CONT_FSM_B,
483                CS_FROM_CONT_FSM_C, CS_FROM_CONT_WR_FSM, CS_FROM_CONT_RD_FSM,
484                CS_FROM_XMT_FSM, CS_FROM_RCV_FSM, CS_FROM_CMD_CTL, CS_SELECT,
485                CS_OUT, DEBUG_MODE, DATA_CRC_MSB, CMD_CRC_MSB, DATA_SR_MSB,
486                CMD_SR_MSB, OUT_CYCLE_IS_DATA,
487                SPI_DATA_LINE_ACTIVE, PAR_RESET, SR_3);
488
489 `include "param.v"
490 input CLK;
491 input RESET_IN;
492 input GO_IDLE;
493 input SPI_MODE;
494 input SPI_CSB;
495 input CMD_IN;
496 input CMD_OUT_WO_SPI;
497 input DATA_IN;
498 input DATA_OUT_WO_SPI;
499 input CMD_OE;
500 input DATA_OE;
501 input SPI_DATA_CYCLE;
502 input DATA_CRC_MSB;
503 input CMD_CRC_MSB;
504 input DATA_SR_MSB;
505 input CMD_SR_MSB;
506 input OUT_CYCLE_IS_DATA;
507 input SPI_DATA_LINE_ACTIVE;
508
509 input [2:0]CS_SELECT;
510 input [4:0]CS_FROM_CONT_FSM_A;
511 input [4:0]CS_FROM_CONT_FSM_B;
512 input [5:0]CS_FROM_CONT_FSM_C;
513 input [3:0]CS_FROM_CONT_WR_FSM;
514 input [4:0]CS_FROM_CONT_RD_FSM;
515 input [5:0]CS_FROM_XMT_FSM;
516 input [5:0]CS_FROM_RCV_FSM;
517 input [5:0]CS_FROM_CMD_CTL;
518 input DEBUG_MODE;
519
520 output RESET;
521 output SPI_CYCLE_5;
522 output SPI_CYCLE_6;
523 output CMD_IN_W_SPI;
524 output CMD_OUT;
525 output DATA_IN_W_SPI;
526 output DATA_OUT;
527 output PAR_RESET;
528 output [5:0]CS_OUT;
529 output SR_3;
530

```

```

531
532 reg [5:0]CS_OUT;
533 reg RESET;
534 reg PAR_RESET;
535 reg SR_1;
536 reg SR_2;
537 reg SR_3;
538
539
540 reg SPI_CYCLE_5;
541 reg SPI_CYCLE_6;
542 reg [2:0]SPI_CYCLE_CNT;
543 reg TMP_CSB_FALLING_EDGE;
544 wire CSB_FALLING_EDGE;
545
546 reg CMD_IN_W_SPI;
547 reg INT_CMD_OUT;
548 reg CMD_OUT;
549 reg DATA_IN_W_SPI;
550 reg INT_DATA_OUT;
551 reg DATA_OUT;
552 wire BLOCKED_CMD_IN;
553
554
555
556 //=====
557 //mux out the fsm's to increase fault coverage
558 //=====
559 always @(CS_SELECT or CS_FROM_CONT_FSM_A or CS_FROM_CONT_FSM_B or
560 CS_FROM_CONT_FSM_C or CS_FROM_CONT_WR_FSM or CS_FROM_CONT_RD_FSM or
561 CS_FROM_XMT_FSM or CS_FROM_RCV_FSM or CS_FROM_CMD_CTL or
562 DATA_CRC_MSB or CMD_CRC_MSB or DATA_SR_MSB or CMD_SR_MSB or DEBUG_MODE)
563 begin
564     case (CS_SELECT)
565         3'b000 : begin
566             CS_OUT <= {DATA_CRC_MSB, CS_FROM_CONT_FSM_A};
567         end
568         3'b001 : begin
569             CS_OUT <= {CMD_CRC_MSB, CS_FROM_CONT_FSM_B};
570         end
571         3'b010 : begin
572             CS_OUT <= CS_FROM_CONT_FSM_C;
573         end
574         3'b011 : begin
575             CS_OUT <= {DATA_SR_MSB, CMD_SR_MSB, CS_FROM_CONT_WR_FSM};
576         end
577         3'b100 : begin
578             CS_OUT <= {1'b0, CS_FROM_CONT_RD_FSM};
579         end
580         3'b101 : begin
581             CS_OUT <= CS_FROM_XMT_FSM;
582         end
583         3'b110 : begin
584             CS_OUT <= CS_FROM_RCV_FSM;
585         end
586         3'b111 : begin

```

```

587     CS_OUT <= CS_FROM_CMD_CTL;
588 end
589     default : begin
590         CS_OUT <= {1'b0, CS_FROM_CONT_FSM_A};
591     end
592 endcase
593 if (!DEBUG_MODE) CS_OUT <= 0;
594 end
595
596
597
598
599 //=====
600 //lock and synchronize the RESET_IN signal
601 //=====
602 always @(posedge CLK or negedge RESET_IN)
603 begin
604     if (!RESET_IN)
605     begin
606         SR_1 <= 0;
607         SR_2 <= 0;
608         SR_3 <= 0;
609     end
610     else
611     begin
612         SR_1 <= 1;
613         SR_2 <= SR_1;
614         SR_3 <= SR_2;
615     end
616 end
617
618
619
620 //reset
621 always @(SR_3 or GO_IDLE)
622 begin
623     if (!SR_3 || GO_IDLE) RESET <= 0;
624     else RESET <= 1;
625
626     if (!SR_3) PAR_RESET <= 0;
627     else PAR_RESET <= 1;
628
629 end
630
631
632 //=====
633 //SPI
634 //=====
635
636 //spi cycle counter, is a modulo 8 counter
637 always @(posedge CLK)
638 begin
639     if (!PAR_RESET) SPI_CYCLE_CNT <= 0;
640     else if (CSB_FALLING_EDGE) SPI_CYCLE_CNT <= 1;
641     else SPI_CYCLE_CNT <= SPI_CYCLE_CNT + 1;
642 end

```

```

643
644
645
646 //clear spi cycle counter
647 always @(posedge CLK)
648     begin
649         if(!PAR_RESET)
650             TMP_CSB_FALLING_EDGE <= 0;
651         else TMP_CSB_FALLING_EDGE <= ~SPI_CSB;
652     end
653 assign CSB_FALLING_EDGE = ~(TMP_CSB_FALLING_EDGE | SPI_CSB);
654
655
656
657 //spi 7, 8 cycle detector
658 always @(SPI_CYCLE_CNT or SPI_CSB or SPI_MODE)
659     begin
660         SPI_CYCLE_5 <= 0;
661         SPI_CYCLE_6 <= 0;
662         if((!SPI_CSB && SPI_MODE) || !SPI_MODE)
663             begin
664                 //if (SPI_CYCLE_CNT == 3'h3) SPI_CYCLE_5 <= 1;
665                 //if (SPI_CYCLE_CNT == 3'h4) SPI_CYCLE_6 <= 1;
666
667                 if (SPI_CYCLE_CNT == 3'h4) SPI_CYCLE_5 <= 1;
668                 if (SPI_CYCLE_CNT == 3'h5) SPI_CYCLE_6 <= 1;
669
670             end
671     end
672
673 //=====
674 //OUT/IN MUX
675 //=====
676
677
678 //if the data line is beeing driven then block the command line
679 assign BLOCKED_CMD_IN = CMD_IN | SPI_DATA_LINE_ACTIVE;
680
681
682 always @(SPI_MODE or CMD_IN or CMD_OUT_WO_SPI or DATA_IN or DATA_OUT_WO_SPI
683         or CMD_OE or DATA_OE or SPI_DATA_CYCLE or OUT_CYCLE_IS_DATA or
684         SPI_CSB or BLOCKED_CMD_IN or
685         SPI_DATA_LINE_ACTIVE)
686     begin
687         INT_CMD_OUT <= 0;
688         if(!SPI_MODE)
689             begin
690                 CMD_IN_W_SPI <= CMD_IN;
691                 INT_CMD_OUT <= CMD_OUT_WO_SPI;
692                 DATA_IN_W_SPI <= DATA_IN;
693                 INT_DATA_OUT <= DATA_OUT_WO_SPI;
694             end
695         else
696             begin
697                 if (SPI_CSB)
698                     begin

```

```

699     CMD_IN_W_SPI <= 1;
700     DATA_IN_W_SPI <= 1;
701 end
702 else if (SPI_DATA_CYCLE)
703 begin
704     CMD_IN_W_SPI <= 1;
705     DATA_IN_W_SPI <= BLOCKED_CMD_IN;
706 end
707 else
708 begin
709     CMD_IN_W_SPI <= BLOCKED_CMD_IN;
710     DATA_IN_W_SPI <= 1;
711 end
712 if (SPI_DATA_LINE_ACTIVE)
713 begin
714     if (!OUT_CYCLE_IS_DATA) INT_DATA_OUT <= CMD_OUT_WO_SPI;
715     else INT_DATA_OUT <= DATA_OUT_WO_SPI;
716 end
717 else INT_DATA_OUT <= 1;
718 end
719 end
720
721
722 //Latching DATA_OUT on the falling edge of the clock
723 always @(negedge CLK)
724 begin
725     if (!RESET) DATA_OUT <= 1;
726     else DATA_OUT <= INT_DATA_OUT;
727 end
728
729 //Latching CMD_OUT on the falling edge of the clock
730 always @(negedge CLK or negedge RESET)
731 begin
732     if (!RESET) CMD_OUT <= 1;
733     else CMD_OUT <= INT_CMD_OUT;
734 end
735
736
737
738 endmodule // top_lgc

```

```

739      3  cmd_unit
740  //-----|
741  module cmd_unit (CLK, RESET, CMD, CMD_IN, CMD_OUT, BE_ERROR_BUS, DATA_OUT,
742  CID_DATA, LD_BLOCKLEN_REG, LD_RCA_REG, LD_DSR_REG, SET_MP_INT,
743  CLR_STOP_TRANSMISSION_FROM_CONT_UNIT, CLR_STOP_TRANSMISSION_FROM_DATA_UNIT,
744  CID_MODE, WP_MODE, SEND_CID_RESPONSE, SEND_OCR_RESPONSE, SEND_RESPONSE,
745  BLOCKLEN_REG, BE_CMD_REG, RCA_MATCH, SECTOR_ADDR_REG,
746  BLOCKLEN_OK, CMD_BUS, PRG_CMD, TAG_CMD, FLASH_WR_CMD, DATA_CMD,
747  STOP_TRANSMISSION, RESPONSE_DONE, START_ADDRESS_OK, GO_IDLE,
748  GLOBAL_CMD,
749  SET_OCR_BUSY, CLR_OCR_BUSY, SET_BLOCK_LEN_ERROR, SET_ERASE_RESET,
750  SET_ERASE_SEQ_ERROR, LD_ADDR_STATUS_BITS, LD_BE_ERROR_REG,
751  SET_ADDRESS_ERROR_FROM_DATA_UNIT, SET_ADDRESS_ERROR_FROM_CONT_UNIT,
752  SET_UNDERRUN_ERROR, SET_OVERRUN_ERROR, CURRENT_STATE, COMPATIBLE, OCR_IN_IS_0,
753  ARG_8_TO_0, CLR_CID_CHECK, SET_CID_CHECK, CID_ERROR, SELECT_CARD_IF_RCA_MATCH,
754  SELECT_CARD_IF_NOT_RCA_MATCH, RESPONSE_START, RESPONSE_DONE_FROM_DATA_UNIT,
755  SEND_RESPONSE_FROM_CONT_UNIT, LOCK_CMD_BUS, LOCKED_CMD_BUS, SET_CMD_DIR_OUT,
756  LD_END_ON_DATA, HOLD_XMT_FSM, DATA_OE, CLR_DATA_OE, SET_DATA_OE,
757  SET_BUSY_FROM_DATA_UNIT, CLR_BUSY_FROM_DATA_UNIT, CLR_BUSY_FROM_CONT_UNIT,
758  BUSY_FLAG, ALL_CLR_BUSY, READY_TO_ERASE, CLR_MAIN_OE, SET_MAIN_OE, CLR_OC_PP,
759  SET_OC_PP, BWR_IS_1, SET_BWR_ERROR, STRM, ANY_BLK_LEN, GO_TEST, TEST_MODE,
760  TEST_CMD, SET_CID_OVERWRITE, ARG_24_TO_9, SPI_MODE, SPI_CYCLE_6,
761  SEND_RESPONSE_SPI_BE, LD_CHECK_CRC_REG, CHECK_FOR_CRC, SPI_DATA_CYCLE,
762  CS_FROM_CMD_CTL, CMD_CRC_MSB, CMD_SR_MSB, BLOCKLEN_OK_FOR_WRITE, PAR_RESET, SR_3,
763  SPI_CSB);
764
765  `include "param.v"
766  //input
767  input CLK;
768  input RESET;
769  input CMD_IN;
770  input [7:0]BE_ERROR_BUS;
771  input DATA_OUT;
772  input CID_DATA;
773  input LD_BLOCKLEN_REG;
774  input LD_RCA_REG;
775  input LD_DSR_REG;
776  input SET_MP_INT;
777  input CLR_STOP_TRANSMISSION_FROM_CONT_UNIT;
778  input CLR_STOP_TRANSMISSION_FROM_DATA_UNIT;
779  input CID_MODE;
780  input WP_MODE;
781  input SEND_CID_RESPONSE;
782  input SEND_OCR_RESPONSE;
783  input SEND_RESPONSE;
784  input SET_OCR_BUSY;
785  input CLR_OCR_BUSY;
786  input SET_BLOCK_LEN_ERROR;
787  input SET_ERASE_RESET;
788  input SET_ERASE_SEQ_ERROR;
789  input LD_ADDR_STATUS_BITS;
790  input LD_BE_ERROR_REG;
791  input SET_ADDRESS_ERROR_FROM_DATA_UNIT;
792  input SET_ADDRESS_ERROR_FROM_CONT_UNIT;
793  input SET_UNDERRUN_ERROR;

```

```

794 input SET_OVERRUN_ERROR;
795 input [3:0]CURRENT_STATE;
796 input CLR_CID_CHECK;
797 input SET_CID_CHECK;
798 input SELECT_CARD_IF_RCA_MATCH;
799 input SELECT_CARD_IF_NOT_RCA_MATCH;
800 input RESPONSE_DONE_FROM_DATA_UNIT;
801 input SEND_RESPONSE_FROM_CONT_UNIT;
802 input LOCK_CMD_BUS;
803 input DATA_OE;
804 input SET_BUSY_FROM_DATA_UNIT;
805 input CLR_BUSY_FROM_DATA_UNIT;
806 input CLR_BUSY_FROM_CONT_UNIT;
807 input READY_TO_ERASE;
808 input SET_BWR_ERROR;
809 input STRM;
810 input ANY_BLK_LEN;
811 input TEST_MODE;
812 input SET_CID_OVERWRITE;
813 output SPI_MODE;
814 input SPI_CYCLE_6;
815 input SEND_RESPONSE_SPI_BE;
816 input LD_CHECK_CRC_REG;
817 input SPI_DATA_CYCLE;
818 input PAR_RESET;
819 input SR_3;
820 input SPI_CSB;
821
822
823 output CMD;
824 output CMD_OUT;
825 output [9:0]BLOCKLEN_REG;
826 output [5:0]BE_CMD_REG;
827 output RCA_MATCH;
828 output [15:0]SECTOR_ADDR_REG;
829 output BLOCKLEN_OK;
830 output [39:0]CMD_BUS;
831 output PRG_CMD;
832 output TAG_CMD;
833 output FLASH_WR_CMD;
834 output DATA_CMD;
835 output STOP_TRANSMISSION;
836 output RESPONSE_DONE;
837 output START_ADDRESS_OK;
838 output GO_IDLE;
839 output GLOBAL_CMD;
840 output COMPATIBLE;
841 output OCR_IN_IS_0;
842 output [8:0]ARG_8_TO_0;
843 output CID_ERROR;
844 output RESPONSE_START;
845 output [10:0]LOCKED_CMD_BUS;
846 output SET_CMD_DIR_OUT;
847 output LD_END_ON_DATA;
848 output HOLD_XMT_FSM;
849 output CLR_DATA_OE;

```



```

850 output SET_DATA_OE;
851 output BUSY_FLAG;
852 output ALL_CLR_BUSY;
853 output CLR_MAIN_OE;
854 output SET_MAIN_OE;
855 output CLR_OC_PP;
856 output SET_OC_PP;
857 output BWR_IS_1;
858 output GO_TEST;
859 output TEST_CMD;
860 output [15:0]ARG_24_TO_9;
861 output CHECK_FOR_CRC;
862 output CMD_CRC_MSB;
863 output CMD_SR_MSB;
864 output BLOCKLEN_OK_FOR_WRITE;
865 output [5:0]CS_FROM_CMD_CTL;
866
867 //cmd_sr
868
869 //cmd_lgc
870 wire FLASH_RD_CMD;
871
872 //cmd_ctl
873
874 //cmd_response
875
876
877 //output
878 //cmd_sr
879 wire [44:0]SR_OUT;
880 wire START_BIT;
881 wire HOST_CARD;
882 wire RESPONSE_BIT;
883 wire CRC_IN_TO_HOST;
884
885
886
887 //cmd_lgc
888
889 wire [5:0] CMD_REG;
890 wire OK_TO_RESPONSE;
891 wire CMD_OK_FOR_STATE;
892 wire STOP_WRITE;
893 wire STOP_READ;
894 wire ERASE_WP;
895
896
897
898 wire DIR_OUT;
899 wire CMD_CNT_DONE;
900 wire [31:0]STATUS_REG;
901 wire [31:0]OCR_REG;
902 wire IGNORE_SELECT_CARD;
903 wire READY_TO_DESELECT;
904 wire READY_TO_SELECT;
905 wire READY_TO_PP;

```

```

906 wire READY_TO_OC;
907 wire IGNORE_ILLEGAL_COMMAND;
908
909
910
911
912 //cmd_ctl
913 wire LD_CMD_CNT_START_DELAY;
914 wire LD_CMD_CNT_CMD_ARG_LEN;
915 wire LD_CMD_CNT_LONG_RESPONSE_LEN;
916 wire CTL_LD_CMD_CNT_CRC_LEN;
917 wire CTL_EN_CMD_CNT;
918 wire CTL_SR_EN;
919 wire CLR_CRC_FROM_CMD_CTL;
920 wire CRC_EN_FROM_CMD_CTL;
921 wire CLR_DIR_OUT;
922 wire SET_COM_CRC_ERROR;
923 wire SET_STOP_TRANSMISSION;
924 wire LD_ARG_REG;
925 wire LOCK_PRE_CMD;
926 wire SET_SR_LSB;
927 wire RESPONSE_CYCLE;
928 wire SET_SR_MSB;
929 wire SEND_RESPONSE_FROM_CMD_CTL;
930 wire SET_BUSY_FROM_CMD_CTL;
931 wire SET_ILLEGAL_COMMAND;
932 wire SEND_SPI_RESPONSE_SHORT;
933
934 //cmd_response
935 wire LD_STATUS;
936 wire LD_CRC;
937 wire LD_STATUS_HEAD;
938 wire LD_END_BIT;
939 wire LD_CID_1;
940 wire LD_OCR_STATUS;
941 wire SET_DIR_OUT;
942 wire RES_EN_CMD_CNT;
943 wire RES_SR_EN;
944 wire LD_CMD_CNT_ARG_LEN;
945 wire LD_CMD_CNT_RESPONSE_HEAD_LEN;
946 wire RES_LD_CMD_CNT_CRC_LEN;
947 wire CLR_STATUS;
948 wire LD_OCR_END;
949 wire CLR_CRC_FROM_CMD_RESPONSE;
950 wire CRC_EN_FROM_CMD_RESPONSE;
951 wire LD_OCR_STATUS_HEAD;
952 wire LD_SPI_RESPONSE;
953 wire LD_SPI_BE_RESPONSE;
954
955 //cmd_crc
956 wire [6:0]CRC_BUS;
957 wire CRC_OK;
958
959
960
961

```

```

962
963
964 cmd_sr u_cmd_sr(.CLK(CLK), .RESET(RESET), .CMD_IN(CMD_IN), .SR_OUT(SR_OUT),
965 .CTL_SR_EN(CTL_SR_EN), .RES_SR_EN(RES_SR_EN), .DIR_OUT(DIR_OUT),
966 .CMD_OUT(CMD_OUT),
967 .START_BIT(START_BIT), .HOST_CARD(HOST_CARD), .RESPONSE_BIT(RESPONSE_BIT),
968 .CID_DATA(CID_DATA), .DATA_OUT(DATA_OUT),
969 .LD_CRC(LD_CRC), .LD_STATUS_HEAD(LD_STATUS_HEAD), .LD_END_BIT(LD_END_BIT),
970 .LD_CID_1(LD_CID_1), .LD_OCR_STATUS(LD_OCR_STATUS), .LD_STATUS(LD_STATUS),
971 .CRC_BUS(CRC_BUS), .CMD_REG(CMD_REG), .OCR_REG(OCR_REG),
972 .STATUS_REG(STATUS_REG), .LD_OCR_END(LD_OCR_END),
973 .CLR_CID_CHECK(CLR_CID_CHECK), .SET_CID_CHECK(SET_CID_CHECK),
974 .CID_ERROR(CID_ERROR), .SET_SR_LSB(SET_SR_LSB),
975 .CRC_IN_TO_HOST(CRC_IN_TO_HOST),
976 .SEND_RESPONSE_FROM_CONT_UNIT(SEND_RESPONSE_FROM_CONT_UNIT),
977 .SET_SR_MSB(SET_SR_MSB), .LD_OCR_STATUS_HEAD(LD_OCR_STATUS_HEAD),
978 .LD_SPI_RESPONSE(LD_SPI_RESPONSE), .LD_SPI_BE_RESPONSE(LD_SPI_BE_RESPONSE),
979 .CMD_SR_MSB(CMD_SR_MSB));
980
981 cmd_lgc u_cmd_lgc(.CLK(CLK), .RESET(RESET), .SR_OUT(SR_OUT[38:7]),
982 .LD_ARG_REG(LD_ARG_REG), .LD_BLOCKLEN_REG(LD_BLOCKLEN_REG),
983 .LD_RCA_REG(LD_RCA_REG),
984 .LD_DSR_REG(LD_DSR_REG), .SET_MP_INT(SET_MP_INT), .ARG_8_TO_0(ARG_8_TO_0),
985 .BLOCKLEN_REG(BLOCKLEN_REG), .CMD_REG(CMD_REG),
986 .BE_CMD_REG(BE_CMD_REG), .RCA_MATCH(RCA_MATCH),
987 .SECTOR_ADDR_REG(SECTOR_ADDR_REG), .BLOCKLEN_OK(BLOCKLEN_OK),
988 .CMD_BUS(CMD_BUS),
989 .PRG_CMD(PRG_CMD), .TAG_CMD(TAG_CMD), .FLASH_WR_CMD(FLASH_WR_CMD),
990 .FLASH_RD_CMD(FLASH_RD_CMD), .DATA_CMD(DATA_CMD),
991 .STOP_TRANSMISSION(STOP_TRANSMISSION),
992 .SET_STOP_TRANSMISSION(SET_STOP_TRANSMISSION),
993 .CLR_STOP_TRANSMISSION_FROM_CONT_UNIT(CLR_STOP_TRANSMISSION_FROM_CONT_UNIT),
994 .CLR_STOP_TRANSMISSION_FROM_DATA_UNIT(CLR_STOP_TRANSMISSION_FROM_DATA_UNIT),
995 .DIR_OUT(DIR_OUT),
996 .SET_DIR_OUT(SET_DIR_OUT), .CLR_DIR_OUT(CLR_DIR_OUT),
997 .LD_CMD_CNT_START_DELAY(LD_CMD_CNT_START_DELAY),
998 .RES_EN_CMD_CNT(RES_EN_CMD_CNT), .CTL_EN_CMD_CNT(CTL_EN_CMD_CNT),
999 .CMD_CNT_DONE(CMD_CNT_DONE), .START_ADDRESS_OK(START_ADDRESS_OK),
1000 .LD_CMD_CNT_CMD_ARG_LEN(LD_CMD_CNT_CMD_ARG_LEN),
1001 .LD_CMD_CNT_LONG_RESPONSE_LEN(LD_CMD_CNT_LONG_RESPONSE_LEN),
1002 .CTL_LD_CMD_CNT_CRC_LEN(CTL_LD_CMD_CNT_CRC_LEN),
1003 .RES_LD_CMD_CNT_CRC_LEN(RES_LD_CMD_CNT_CRC_LEN), .CID_MODE(CID_MODE),
1004 .WP_MODE(WP_MODE), .LD_CMD_CNT_ARG_LEN(LD_CMD_CNT_ARG_LEN),
1005 .LD_CMD_CNT_RESPONSE_HEAD_LEN(LD_CMD_CNT_RESPONSE_HEAD_LEN),
1006 .SET_OCR_BUSY(SET_OCR_BUSY), .CLR_OCR_BUSY(CLR_OCR_BUSY),
1007 .SET_BLOCK_LEN_ERROR(SET_BLOCK_LEN_ERROR), .SET_ERASE_RESET(SET_ERASE_RESET),
1008 .SET_ERASE_SEQ_ERROR(SET_ERASE_SEQ_ERROR),
1009 .LD_ADDR_STATUS_BITS(LD_ADDR_STATUS_BITS), .LD_BE_ERROR_REG(LD_BE_ERROR_REG),
1010 .SET_ADDRESS_ERROR_FROM_DATA_UNIT(SET_ADDRESS_ERROR_FROM_DATA_UNIT),
1011 .SET_ADDRESS_ERROR_FROM_CONT_UNIT(SET_ADDRESS_ERROR_FROM_CONT_UNIT),
1012 .SET_UNDERRUN_ERROR(SET_UNDERRUN_ERROR),
1013 .SET_OVERRUN_ERROR(SET_OVERRUN_ERROR),
1014 .CURRENT_STATE(CURRENT_STATE), .SET_ILLEGAL_COMMAND(SET_ILLEGAL_COMMAND),
1015 .SET_CID_OVERWRITE(SET_CID_OVERWRITE), .BE_ERROR_BUS(BE_ERROR_BUS),
1016 .SET_COM_CRC_ERROR(SET_COM_CRC_ERROR), .CLR_STATUS(CLR_STATUS),
1017 .STATUS_REG(STATUS_REG), .OCR_REG(OCR_REG), .SEND_RESPONSE(SEND_RESPONSE),

```

```

1018 .COMPATIBLE(COMPATIBLE), .OCR_IN_IS_0(OCR_IN_IS_0), .LOCK_PRE_CMD(LOCK_PRE_CMD),
1019 .SELECT_CARD IF RCA_MATCH(SELECT_CARD IF RCA_MATCH),
1020 .SELECT_CARD IF NOT RCA_MATCH(SELECT_CARD IF NOT RCA_MATCH),
1021 .IGNORE_SELECT_CARD(IGNORE_SELECT_CARD), .LOCK_CMD_BUS(LOCK_CMD_BUS),
1022 .LOCKED_CMD_BUS(LOCKED_CMD_BUS), .OK_TO_RESPONSE(OK_TO_RESPONSE),
1023 .SEND_RESPONSE_FROM_CMD_CTL(SEND_RESPONSE_FROM_CMD_CTL),
1024 .SET_BUSY_FROM_DATA_UNIT(SET_BUSY_FROM_DATA_UNIT),
1025 .CLR_BUSY_FROM_DATA_UNIT(CLR_BUSY_FROM_DATA_UNIT),
1026 .CLR_BUSY_FROM_CONT_UNIT(CLR_BUSY_FROM_CONT_UNIT),
1027 .SET_BUSY_FROM_CMD_CTL(SET_BUSY_FROM_CMD_CTL), .BUSY_FLAG(BUSY_FLAG),
1028 .ALL_CLR_BUSY(ALL_CLR_BUSY), .CMD_OK_FOR_STATE(CMD_OK_FOR_STATE),
1029 .STOP_WRITE(STOP_WRITE), .STOP_READ(STOP_READ),
1030 .READY_TO_ERASE(READY_TO_ERASE), .ERASE_WP(ERASE_WP),
1031 .READY_TO_DESELECT(READY_TO_DESELECT), .READY_TO_SELECT(READY_TO_SELECT),
1032 .READY_TO_PP(READY_TO_PP), .READY_TO_OC(READY_TO_OC), .BWR_IS_1(BWR_IS_1),
1033 .SET_BWR_ERROR(SET_BWR_ERROR), .STRM(STRM), .ANY_BLK_LEN(ANY_BLK_LEN),
1034 .IGNORE_ILLEGAL_COMMAND(IGNORE_ILLEGAL_COMMAND), .TEST_CMD(TEST_CMD),
1035 .TEST_MODE(TEST_MODE), .ARG_24_TO_9(ARG_24_TO_9), .SPI_MODE(SPI_MODE),
1036 .LD_CHECK_CRC_REG(LD_CHECK_CRC_REG), .CHECK_FOR_CRC(CHECK_FOR_CRC),
1037 .BLOCKLEN_OK_FOR_WRITE(BLOCKLEN_OK_FOR_WRITE));
1038
1039 cmd_ctl u_cmd_ctl(.CLK(CLK), .RESET(RESET),
1040 .LD_CMD_CNT_START_DELAY(LD_CMD_CNT_START_DELAY),
1041 .LD_CMD_CNT_CMD_ARG_LEN(LD_CMD_CNT_CMD_ARG_LEN),
1042 .LD_CMD_CNT_LONG_RESPONSE_LEN(LD_CMD_CNT_LONG_RESPONSE_LEN),
1043 .LD_CMD_CNT_CRC_LEN(CTL_LD_CMD_CNT_CRC_LEN),
1044 .SEND_RESPONSE_FROM_CONT_UNIT(SEND_RESPONSE_FROM_CONT_UNIT),
1045 .EN_CMD_CNT(CTL_EN_CMD_CNT), .CMD_CNT_DONE(CMD_CNT_DONE),
1046 .START_BIT(START_BIT), .SR_EN(CTL_SR_EN), .CLR_CRC(CLR_CRC_FROM_CMD_CTL),
1047 .HOST_CARD(HOST_CARD), .RESPONSE_BIT(RESPONSE_BIT),
1048 .CRC_EN(CRC_EN_FROM_CMD_CTL), .CLR_DIR_OUT(CLR_DIR_OUT),
1049 .RESPONSE_DONE(RESPONSE_DONE), .CRC_OK(CRC_OK),
1050 .SET_COM_CRC_ERROR(SET_COM_CRC_ERROR),
1051 .SET_STOP_TRANSMISSION(SET_STOP_TRANSMISSION), .CMD(CMD), .GO_IDLE(GO_IDLE),
1052 .GLOBAL_CMD(GLOBAL_CMD), .LD_ARG_REG(LD_ARG_REG), .LOCK_PRE_CMD(LOCK_PRE_CMD),
1053 .IGNORE_SELECT_CARD(IGNORE_SELECT_CARD), .SET_SR_LSB(SET_SR_LSB),
1054 .RESPONSE_DONE_FROM_DATA_UNIT(RESPONSE_DONE_FROM_DATA_UNIT),
1055 .RESPONSE_CYCLE(RESPONSE_CYCLE), .CMD_BUS(CMD_BUS),
1056 .SET_CMD_DIR_OUT(SET_CMD_DIR_OUT), .OK_TO_RESPONSE(OK_TO_RESPONSE),
1057 .SET_SR_MSB(SET_SR_MSB), .RCA_MATCH(RCA_MATCH),
1058 .SEND_RESPONSE(SEND_RESPONSE_FROM_CMD_CTL), .LD_END_ON_DATA(LD_END_ON_DATA),
1059 .HOLD_XMT_FSM(HOLD_XMT_FSM), .DATA_OE(DATA_OE), .CLR_DATA_OE(CLR_DATA_OE),
1060 .SET_DATA_OE(SET_DATA_OE), .SET_BUSY(SET_BUSY_FROM_CMD_CTL),
1061 .ERASE_WP(ERASE_WP), .SET_ILLEGAL_COMMAND(SET_ILLEGAL_COMMAND),
1062 .CMD_OK FOR STATE(CMD_OK FOR STATE), .STOP_WRITE(STOP_WRITE),
1063 .STOP_READ(STOP_READ), .FLASH_RD_CMD(FLASH_RD_CMD),
1064 .READY_TO_DESELECT(READY_TO_DESELECT), .READY_TO_SELECT(READY_TO_SELECT),
1065 .CLR_MAIN_OE(CLR_MAIN_OE), .SET_MAIN_OE(SET_MAIN_OE),
1066 .READY_TO_PP(READY_TO_PP), .READY_TO_OC(READY_TO_OC), .CLR_OC_PP(CLR_OC_PP),
1067 .SET_OC_PP(SET_OC_PP), .IGNORE_ILLEGAL_COMMAND(IGNORE_ILLEGAL_COMMAND),
1068 .TEST_CMD(TEST_CMD), .GO_TEST(GO_TEST), .SPI_MODE(SPI_MODE),
1069 .SPI_DATA_CYCLE(SPI_DATA_CYCLE),
1070 .CS(CS_FROM_CMD_CTL), .SEND_SPI_RESPONSE_SHORT(SEND_SPI_RESPONSE_SHORT),
1071 .PAR_RESET(PAR_RESET), .SR_3(SR_3), .SPI_CSB(SPI_CSB));
1072
1073 cmd_response u_cmd_response(.CLK(CLK), .RESET(RESET),

```

```

1074 .SEND_CID_RESPONSE(SEND_CID_RESPONSE), .SEND_OCR_RESPONSE(SEND_OCR_RESPONSE),
1075 .SEND_RESPONSE(SEND_RESPONSE), .SET_DIR_OUT(SET_DIR_OUT), .LD_CID_1(LD_CID_1),
1076 .LD_STATUS_HEAD(LD_STATUS_HEAD),
1077 .CLR_STATUS(CLR_STATUS),
1078 .EN_CMD_CNT(RES_EN_CMD_CNT),
1079 .LD_CMD_CNT_RESPONSE_HEAD_LEN(LD_CMD_CNT_RESPONSE_HEAD_LEN),
1080 .CMD_CNT_DONE(CMD_CNT_DONE), .LD_CMD_CNT_ARG_LEN(LD_CMD_CNT_ARG_LEN),
1081 .LD_STATUS(LD_STATUS), SR_EN(RES_SR_EN), .LD_CRC(LD_CRC),
1082 .LD_CMD_CNT_CRC_LEN(RES_LD_CMD_CNT_CRC_LEN), .LD_END_BIT(LD_END_BIT),
1083 .LD_OCR_STATUS(LD_OCR_STATUS), .LD_OCR_END(LD_OCR_END),
1084 .RESPONSE_START(RESPONSE_START), .RESPONSE_DONE(RESPONSE_DONE),
1085 .CRC_EN(CRC_EN_FROM_CMD_RESPONSE), .CLR_CRC(CLR_CRC_FROM_CMD_RESPONSE),
1086 .SEND_RESPONSE_FROM_CMD_CTL(SEND_RESPONSE_FROM_CMD_CTL),
1087 .LD_OCR_STATUS_HEAD(LD_OCR_STATUS_HEAD), .SPI_CYCLE_6(SPI_CYCLE_6),
1088 .LD_SPI_RESPONSE(LD_SPI_RESPONSE), .SPI_MODE(SPI_MODE), .CMD_BUS(CMD_BUS),
1089 .LD_SPI_BE_RESPONSE(LD_SPI_BE_RESPONSE),
1090 .SEND_RESPONSE_SPI_BE(SEND_RESPONSE_SPI_BE),
1091 .SEND_SPI_RESPONSE_SHORT(SEND_SPI_RESPONSE_SHORT));
1092
1093 cmd_crc u_cmd_crc(.CLK(CLK), .RESET(RESET), .CRC_OK(CRC_OK),
1094 .CLR_CRC_FROM_CMD_RESPONSE(CLR_CRC_FROM_CMD_RESPONSE),
1095 .CLR_CRC_FROM_CMD_CTL(CLR_CRC_FROM_CMD_CTL),
1096 .CRC_EN_FROM_CMD_RESPONSE(CRC_EN_FROM_CMD_RESPONSE),
1097 .CRC_EN_FROM_CMD_CTL(CRC_EN_FROM_CMD_CTL), .CRC_IN_FROM_HOST(CMD_IN),
1098 .CRC_IN_TO_HOST(CRC_IN_TO_HOST), .CRC_BUS(CRC_BUS),
1099 .RESPONSE_CYCLE(RESPONSE_CYCLE), .CHECK_FOR_CRC(CHECK_FOR_CRC),
1100 .CMD_CRC_MSB(CMD_CRC_MSB));
1101
1102 endmodule // cmd_unit

```

```

1103         4 cmd_ctl
1104 //-----|
1105 module cmd_ctl(CLK, RESET, LD_CMD_CNT_START_DELAY, LD_CMD_CNT_CMD_ARG_LEN,
1106 LD_CMD_CNT_LONG_RESPONSE_LEN,
1107 LD_CMD_CNT_CRC_LEN, SEND_RESPONSE_FROM_CONT_UNIT, EN_CMD_CNT,
1108 CMD_CNT_DONE,
1109 START_BIT,
1110 SR_EN, CLR_CRC, HOST_CARD, RESPONSE_BIT, CRC_EN, CLR_DIR_OUT,
1111 RESPONSE_DONE,
1112 CRC_OK, SET_COM_CRC_ERROR, SET_STOP_TRANSMISSION, CMD,
1113 GO_IDLE, GLOBAL_CMD, LD_ARG_REG, LOCK_PRE_CMD,
1114 IGNORE_SELECT_CARD, SET_SR_LSB, RESPONSE_DONE_FROM_DATA_UNIT, RESPONSE_CYCLE,
1115 CMD_BUS, SET_CMD_DIR_OUT, OK_TO_RESPONSE, SET_SR_MSB, RCA_MATCH, SEND_RESPONSE,
1116 LD_END_ON_DATA, HOLD_XMT_FSM, DATA_OE, CLR_DATA_OE, SET_DATA_OE, SET_BUSY,
1117 ERASE_WP, SET_ILLEGAL_COMMAND, CMD_OK_FOR_STATE, STOP_WRITE, STOP_READ,
1118 FLASH_RD_CMD, READY_TO_DESELECT, READY_TO_SELECT, CLR_MAIN_OE, SET_MAIN_OE,
1119 READY_TO_PP, READY_TO_OC, CLR_OC_PP, SET_OC_PP, IGNORE_ILLEGAL_COMMAND,
1120 GO_TEST, TEST_CMD, SPI_MODE, SPI_DATA_CYCLE, CS, SEND_SPI_RESPONSE_SHORT,
1121 PAR_RESET, SR_3, SPI_CSB);
1122
1123
1124
1125
1126
1127 `include "param.v"
1128 input CLK;
1129 input RESET;
1130
1131 input SEND_RESPONSE_FROM_CONT_UNIT;
1132 input CMD_CNT_DONE;
1133 input START_BIT;
1134 input HOST_CARD;
1135 input RESPONSE_BIT;
1136 input RESPONSE_DONE;
1137 input CRC_OK;
1138 input IGNORE_SELECT_CARD;
1139 input RESPONSE_DONE_FROM_DATA_UNIT;
1140 input [39:0]CMD_BUS;
1141 input OK_TO_RESPONSE;
1142 input RCA_MATCH;
1143 input DATA_OE;
1144 input ERASE_WP;
1145 input CMD_OK_FOR_STATE;
1146 input STOP_WRITE;
1147 input STOP_READ;
1148 input FLASH_RD_CMD;
1149 input READY_TO_DESELECT;
1150 input READY_TO_SELECT;
1151 input READY_TO_PP;
1152 input READY_TO_OC;
1153 input IGNORE_ILLEGAL_COMMAND;
1154 input TEST_CMD;
1155 output SPI_MODE;
1156 input SPI_DATA_CYCLE;
1157 input PAR_RESET;

```

```

1158 input SR_3;
1159 input SPI_CSB;
1160
1161
1162 output LD_CMD_CNT_START_DELAY;
1163 output LD_CMD_CNT_CMD_ARG_LEN;
1164 output LD_CMD_CNT_LONG_RESPONSE_LEN;
1165 output LD_CMD_CNT_CRC_LEN;
1166 output EN_CMD_CNT;
1167 output SR_EN;
1168 output CLR_CRC;
1169 output CRC_EN;
1170 output CLR_DIR_OUT;
1171 output SET_COM_CRC_ERROR;
1172 output SET_STOP_TRANSMISSION;
1173 output CMD;
1174 output GO_IDLE;
1175 output GLOBAL_CMD;
1176 output LD_ARG_REG;
1177 output LOCK_PRE_CMD;
1178 output SET_SR_LSB;
1179 output RESPONSE_CYCLE;
1180 output SET_CMD_DIR_OUT;
1181 output SET_SR_MSB;
1182 output SEND_RESPONSE;
1183 output LD_END_ON_DATA;
1184 output HOLD_XMT_FSM;
1185 output CLR_DATA_OE;
1186 output SET_DATA_OE;
1187 output SET_BUSY;
1188 output SET_ILLEGAL_COMMAND;
1189 output CLR_MAIN_OE;
1190 output SET_MAIN_OE;
1191 output CLR_OC_PP;
1192 output SET_OC_PP;
1193 output GO_TEST;
1194 output SEND_SPI_RESPONSE_SHORT;
1195 output [5:0]CS;
1196
1197 reg LD_CMD_CNT_START_DELAY;
1198 reg LD_CMD_CNT_CMD_ARG_LEN;
1199 reg LD_CMD_CNT_LONG_RESPONSE_LEN;
1200 reg LD_CMD_CNT_CRC_LEN;
1201 reg EN_CMD_CNT;
1202 reg SR_EN;
1203 reg CLR_CRC;
1204 reg CRC_EN;
1205 reg CLR_DIR_OUT;
1206 reg SET_COM_CRC_ERROR;
1207 reg SET_STOP_TRANSMISSION;
1208 reg CMD;
1209 reg PRE_GO_IDLE;
1210 reg GLOBAL_CMD;
1211 reg LD_ARG_REG;
1212 reg LOCK_PRE_CMD;
1213 reg SET_SR_LSB;

```

```

1214 reg RESPONSE_CYCLE;
1215 reg INT_SET_CMD_DIR_OUT;
1216 reg SET_SR_MSB;
1217 reg SEND_RESPONSE;
1218 reg LD_END_ON_DATA;
1219 reg HOLD_XMT_FSM;
1220 reg CLR_DATA_OE;
1221 reg SET_DATA_OE;
1222 reg SET_BUSY;
1223 reg SET_ILLEGAL_COMMAND;
1224 reg CLR_MAIN_OE;
1225 reg SET_MAIN_OE;
1226 reg CLR_OC_PP;
1227 reg SET_OC_PP;
1228 reg GO_TEST;
1229 reg SEND_SPI_RESPONSE_SHORT;
1230
1231 //signal section
1232 reg RESPONSE_LONG;
1233 reg SET_RESPONSE_LONG;
1234 reg CLR_RESPONSE_LONG;
1235 reg GO_IDLE;
1236
1237
1238 parameter [5:0] //synopsys enum state_info
1239 START_S = 6'h0,
1240 WAIT_S = 6'h1,
1241 START_BIT_S = 6'h2,
1242 HOST_S = 6'h3,
1243 ST_ARG_S = 6'h4,
1244 ARG_S = 6'h5,
1245 ST_LONG_S = 6'h6,
1246 CLR_DIR_S = 6'h7,
1247 WAIT_RESPONSE_S = 6'h8,
1248 START_CRC_S = 6'h9,
1249 WAIT_CRC_S = 6'h0a,
1250 NOT_CMD_S = 6'h0b,
1251 CRC_ERROR_S = 6'h0c,
1252 STOP_TRAN_S = 6'h0d,
1253 GO_IDLE_S = 6'h0e,
1254 GO_INACTIVE_S = 6'h0f,
1255 RESPONSE_LONG_S = 6'h10,
1256 CMD_S = 6'h11,
1257 START1_BIT_S = 6'h12,
1258 DLY2_S = 6'h13,
1259 CMD_WAIT_S = 6'h14,
1260 WAIT_END_S = 6'h17,
1261 START2_BIT_S = 6'h18,
1262 CMD1_S = 6'h19,
1263 CMD2_S = 6'h1a,
1264 CMD6_S = 6'h1b,
1265 CMD3_S = 6'h1c,
1266 CMD4_S = 6'h1d,
1267 CMD5_S = 6'h1e,
1268 ERASE1_S = 6'h15,
1269 ERASE2_S = 6'h16,

```



```

1270 ILEGAL_S = 6'h1f,
1271 GO_TEST_S = 6'h20,
1272 GO_TEST1_S = 6'h21,
1273 DLY3_S = 6'h22,
1274 DLY4_S = 6'h23,
1275 SPI_RESPONSE_S = 6'h24,
1276 ST_LONG1_S = 6'h25,
1277 NOT_RESPONSE_LONG_S = 6'h26;
1278
1279
1280
1281 reg [5:0] /* synopsys enum state_info */ CS;
1282 reg [5:0] /* synopsys enum state_info */ NS;
1283
1284 // synopsys state_vector CS
1285
1286 //next state logic
1287 always @(CS or CMD_CNT_DONE or START_BIT or HOST_CARD or RESPONSE_BIT or
1288 RESPONSE_LONG
1289 or RESPONSE_DONE or CRC_OK or CMD_BUS or IGNORE_SELECT_CARD or
1290 RESPONSE_DONE_FROM_DATA_UNIT or RCA_MATCH or DATA_OE or OK_TO_RESPONSE or
1291 ERASE_WP or CMD_OK_FOR_STATE or STOP_WRITE or STOP_READ or FLASH_RD_CMD or
1292 READY_TO_DESELECT or READY_TO_SELECT or READY_TO_PP or READY_TO_OC or
1293 IGNORE_ILLEGAL_COMMAND or TEST_CMD or SPI_MODE or SPI_DATA_CYCLE)
1294 begin
1295 LD_CMD_CNT_START_DELAY <= 0;
1296 LD_CMD_CNT_CMD_ARG_LEN <= 0;
1297 LD_CMD_CNT_LONG_RESPONSE_LEN <= 0;
1298 LD_CMD_CNT_CRC_LEN <= 0;
1299 EN_CMD_CNT <= 0;
1300 SR_EN <= 0;
1301 CLR_CRC <= 0;
1302 CRC_EN <= 0;
1303 CLR_RESPONSE_LONG <= 0;
1304 SET_RESPONSE_LONG <= 0;
1305 CLR_DIR_OUT <= 0;
1306 SET_COM_CRC_ERROR <= 0;
1307 SET_STOP_TRANSMISSION <= 0;
1308 CMD <= 0;
1309 PRE_GO_IDLE <= 0;
1310 GLOBAL_CMD <= 0;
1311 LD_ARG_REG <= 0;
1312 LOCK_PRE_CMD <= 0;
1313 SET_SR_LSB <= 0;
1314 RESPONSE_CYCLE <= 0;
1315 INT_SET_CMD_DIR_OUT <= 0;
1316 SET_SR_MSB <= 0;
1317 SEND_RESPONSE <= 0;
1318 LD_END_ON_DATA <= 0;
1319 HOLD_XMT_FSM <= 0;
1320 SET_DATA_OE <= 0;
1321 CLR_DATA_OE <= 0;
1322 SET_BUSY <= 0;
1323 SET_ILLEGAL_COMMAND <= 0;
1324 CLR_MAIN_OE <= 0;
1325 SET_MAIN_OE <= 0;

```

```

1326 CLR_OC_PP <= 0;
1327 SET_OC_PP <= 0;
1328 GO_TEST <= 0;
1329 SEND_SPI_RESPONSE_SHORT <= 0;
1330
1331 case (CS)
1332     START_S : begin
1333         NS <= WAIT_S;
1334         LD_CMD_CNT_START_DELAY <= 1;
1335     end
1336     WAIT_S : begin
1337         if (CMD_CNT_DONE) NS <= START_BIT_S;
1338         else NS <= WAIT_S;
1339         EN_CMD_CNT <= 1;
1340     end
1341     START2_BIT_S : begin
1342         NS <= START1_BIT_S;
1343     end
1344     START1_BIT_S : begin
1345         NS <= START_BIT_S;
1346     end
1347     START_BIT_S : begin
1348         if (START_BIT) NS <= HOST_S;
1349         else NS <= START_BIT_S;
1350         if (START_BIT) CRC_EN <= 1;
1351         SR_EN <= 1;
1352         CLR_CRC <= 1;
1353         SET_SR_MSB <= 1;
1354     end
1355     HOST_S : begin
1356         if (!RESPONSE_BIT) NS <= ST_ARG_S;
1357         else if (RESPONSE_LONG) NS <= ST_LONG_S;
1358         else NS <= ST_ARG_S;
1359         CRC_EN <= 1;
1360         SR_EN <= 1;
1361     end
1362     ST_ARG_S : begin
1363         NS <= ARG_S;
1364         SR_EN <= 1;
1365         CRC_EN <= 1;
1366         LD_CMD_CNT_CMD_ARG_LEN <= 1;
1367         CLR_RESPONSE_LONG <= 1;
1368     end
1369     ARG_S : begin
1370         if (CMD_CNT_DONE) NS <= START_CRC_S;
1371         else NS <= ARG_S;
1372         SR_EN <= 1;
1373         CRC_EN <= 1;
1374         EN_CMD_CNT <= 1;
1375     end
1376
1377     ST_LONG_S : begin
1378         NS <= ST_LONG1_S;
1379         CLR_RESPONSE_LONG <= 1;
1380         LD_CMD_CNT_LONG_RESPONSE_LEN <= 1;
1381     end

```

```

1382
1383 ST_LONG1_S : begin
1384     if (CMD_CNT_DONE) NS <= START_BIT_S;
1385     else NS <= ST_LONG1_S;
1386     EN_CMD_CNT <= 1;
1387     SET_SR_LSB <= 1;
1388 end
1389
1390
1391 CLR_DIR_S : begin
1392     NS <= START1_BIT_S;
1393     CLR_DIR_OUT <= 1;
1394     CLR_RESPONSE_LONG <= 1;
1395     SET_SR_LSB <= 1;
1396 end
1397 WAIT_RESPONSE_S : begin
1398     if (RESPONSE_DONE || RESPONSE_DONE_FROM_DATA_UNIT) NS <= CLR_DIR_S;
1399     else NS <= WAIT_RESPONSE_S;
1400     RESPONSE_CYCLE <= 1;
1401 end
1402 START_CRC_S : begin
1403     NS <= WAIT_CRC_S;
1404     SR_EN <= 1;
1405     LD_CMD_CNT_CRC_LEN <= 1;
1406     LOCK_PRE_CMD <= 1;
1407     CRC_EN <= 1;
1408 end
1409
1410 WAIT_CRC_S : begin
1411     if (CMD_CNT_DONE) NS <= WAIT_END_S;
1412     else NS <= WAIT_CRC_S;
1413     SR_EN <= 1;
1414     EN_CMD_CNT <= 1;
1415     CRC_EN <= 1;
1416 end //case
1417
1418
1419
1420 WAIT_END_S : begin
1421     if (CRC_OK) begin
1422         if (HOST_CARD) begin
1423             if (CMD_OK_FOR_STATE) begin
1424                 if (STOP_WRITE) NS <= STOP_TRAN_S;
1425                 else if (ERASE_WP) NS <= ERASE1_S;
1426                 else if ('GO_IDLE_STATE) NS <= GO_IDLE_S;
1427                 else if (TEST_CMD) NS <= GO_TEST_S;
1428                 else if ('GO_INACTIVE_STATE || 'SEND_STATUS)
1429                     begin
1430                         if (RCA_MATCH) NS <= GO_INACTIVE_S;
1431                         else NS <= START2_BIT_S;
1432                     end
1433                 else if (IGNORE_SELECT_CARD) NS <= START2_BIT_S;
1434                 else begin
1435                     if ('SELECT_CARD) NS <= CMD5_S;
1436                     else if (FLASH_RD_CMD) NS <= CMD1_S;
1437                     else if (STOP_READ) NS <= CMD3_S;

```

```

1438         else NS <= CMD_S;
1439     end //else
1440 end //CMD_OK_FOR_STATE
1441 else NS <= ILEGAL_S;
1442 end //HOST_CARD
1443     else begin
1444         if (!SPI_MODE) NS <= NOT_CMD_S;
1445         else NS <= ILEGAL_S;
1446     end
1447 end //CRC_OK
1448     else NS <= CRC_ERROR_S;
1449     SET_SR_LSB <= 1;
1450     SET_SR_MSB <= 1;
1451 end //case
1452
1453
1454 ILEGAL_S : begin
1455     if (SPI_MODE)
1456     begin
1457         NS <= SPI_RESPONSE_S;
1458         INT_SET_CMD_DIR_OUT <= 1;
1459     end
1460     else
1461     begin
1462         if ('ALL_SEND_CID) NS <= RESPONSE_LONG_S;
1463         else if ('SEND_CSD || 'SEND_CID)
1464         begin
1465             if (RCA_MATCH) NS <= NOT_RESPONSE_LONG_S;
1466             else NS <= RESPONSE_LONG_S;
1467         end
1468         else NS <= NOT_RESPONSE_LONG_S;
1469     end
1470
1471     if (!IGNORE_ILLEGAL_COMMAND) SET_ILLEGAL_COMMAND <= 1;
1472 end
1473
1474
1475
1476
1477
1478
1479 NOT_CMD_S : begin
1480     NS <= START1_BIT_S;
1481 end
1482
1483 CRC_ERROR_S : begin
1484     if (SPI_MODE)
1485     begin
1486         NS <= SPI_RESPONSE_S;
1487         INT_SET_CMD_DIR_OUT <= 1;
1488     end
1489     else NS <= START1_BIT_S;
1490     SET_COM_CRC_ERROR <= 1;
1491 end
1492
1493

```

```

1494 SPI_RESPONSE_S : begin
1495     NS <= START_BIT_S;
1496     SEND_SPI_RESPONSE_SHORT <= 1;
1497 end
1498
1499
1500
1501
1502 STOP_TRAN_S : begin
1503     NS <= DLY2_S;
1504     SET_STOP_TRANSMISSION <= 1;
1505     INT_SET_CMD_DIR_OUT <= 1;
1506 end
1507
1508 DLY2_S : begin
1509     NS <= DLY3_S;
1510 end
1511
1512 DLY3_S : begin
1513     NS <= DLY4_S;
1514     SET_DATA_OE <= 1;
1515 end
1516
1517 DLY4_S : begin
1518     NS <= START_BIT_S;
1519     SEND_RESPONSE <= 1;
1520     SET_BUSY <= 1;
1521 end
1522
1523
1524
1525 GO_IDLE_S : begin
1526     NS <= START1_BIT_S;
1527     PRE_GO_IDLE <= 1;
1528 end
1529
1530
1531 GO_TEST_S : begin
1532     NS <= GO_TEST1_S;
1533     GO_TEST <= 1;
1534     INT_SET_CMD_DIR_OUT <= 1;
1535 end
1536
1537
1538 GO_TEST1_S : begin
1539     NS <= START_BIT_S;
1540     CLR_OC_PP <= 1;
1541     SET_MAIN_OE <= 1;
1542 end
1543
1544 GO_INACTIVE_S : begin
1545     if (SEND_STATUS) NS <= START1_BIT_S;
1546     else NS <= GO_INACTIVE_S;
1547     GLOBAL_CMD <= 1;
1548 end
1549

```

```

1550 RESPONSE_LONG_S : begin
1551     NS <= START_BIT_S;
1552     SET_RESPONSE_LONG <= 1;
1553     end
1554
1555 NOT_RESPONSE_LONG_S : begin
1556     NS <= START_BIT_S;
1557     CLR_RESPONSE_LONG <= 1;
1558     end
1559
1560
1561 CMD_S : begin
1562     NS <= CMD_WAIT_S;
1563     INT_SET_CMD_DIR_OUT <= 1;
1564     LD_ARG_REG <= 1;
1565     end
1566
1567
1568
1569
1570
1571 CMD_WAIT_S : begin
1572     NS <= START_BIT_S;
1573     CMD <= 1;
1574     if (READY_TO_PP)
1575         begin
1576             CLR_OC_PP <= 1;
1577             SET_MAIN_OE <= 1;
1578         end
1579     if (READY_TO_OC) SET_OC_PP <= 1;
1580     if (!RCA_MATCH && ('SEND_CSD || 'SEND_CID)) SET_RESPONSE_LONG <= 1;
1581     else CLR_RESPONSE_LONG <= 1;
1582     end
1583
1584 CMD1_S : begin
1585     NS <= CMD2_S;
1586     LD_ARG_REG <= 1;
1587     INT_SET_CMD_DIR_OUT <= 1;
1588     SET_DATA_OE <= 1;
1589     LD_END_ON_DATA <= 1;
1590     HOLD_XMT_FSM <= 1;
1591     end
1592
1593 CMD2_S : begin
1594     NS <= START_BIT_S;
1595     //LD_END_ON_DATA <= 1;
1596     //HOLD_XMT_FSM <= 1;
1597     CMD <= 1;
1598     end
1599
1600 CMD3_S : begin
1601     NS <= CMD4_S;
1602     SET_STOP_TRANSMISSION <= 1;
1603     LD_END_ON_DATA <= 1;
1604     HOLD_XMT_FSM <= 1;
1605     INT_SET_CMD_DIR_OUT <= 1;
1606     end
1607
1608 CMD4_S : begin

```

```

1606         NS <= START_BIT_S;
1607         CLR_DATA_OE <= 1;
1608         SEND_RESPONSE <= 1;
1609         end
1610     CMD5_S : begin
1611         NS <= CMD6_S;
1612         if (READY_TO_SELECT) INT_SET_CMD_DIR_OUT <= 1;
1613         end
1614     CMD6_S : begin
1615         NS <= START_BIT_S;
1616         if (READY_TO_DESELECT) CLR_MAIN_OE <= 1;
1617         if (READY_TO_SELECT) SET_MAIN_OE <= 1;
1618         CMD <= 1;
1619         end
1620
1621
1622     ERASE1_S : begin
1623         NS <= ERASE2_S;
1624         LD_ARG_REG <= 1;
1625         SET_DATA_OE <= 1;
1626         INT_SET_CMD_DIR_OUT <= 1;
1627         end
1628     ERASE2_S : begin
1629         NS <= START_BIT_S;
1630         SET_BUSY <= 1;
1631         CMD <= 1;
1632         end
1633
1634
1635
1636         default : begin
1637             NS <= START_S;
1638         end
1639     endcase
1640
1641 end
1642
1643 //current state register
1644 always @(posedge CLK)
1645     begin
1646         if (!PAR_RESET) CS <= START_S;
1647         else if (SEND_RESPONSE_FROM_CONT_UNIT || SEND_RESPONSE ||
1648             SEND_SPI_RESPONSE_SHORT) CS <= WAIT_RESPONSE_S;
1649         else CS <= NS;
1650     end
1651
1652
1653 //clean GO_IDLE from spikes because it is connected
1654 //to async. reset in reg_unit
1655 always @(posedge CLK)
1656     begin
1657         GO_IDLE <= PRE_GO_IDLE;
1658     end
1659
1660
1661

```

```

1662 //response long srff
1663 always @(posedge CLK)
1664 begin
1665     if (RESET == 0) RESPONSE_LONG <= 0;
1666     else if (SET_RESPONSE_LONG) RESPONSE_LONG <= 1;
1667     else if (CLR_RESPONSE_LONG) RESPONSE_LONG <= 0;
1668 end
1669
1670 assign SET_CMD_DIR_OUT = OK_TO_RESPONSE & INT_SET_CMD_DIR_OUT;
1671
1672 // Detect SPI Operation Mode
1673 reg          SPI_MODE;
1674
1675 always @(posedge CLK or negedge SR_3)
1676 begin
1677     if (~SR_3) SPI_MODE <= 0;
1678     else if (PRE_GO_IDLE & ~SPI_CSB) SPI_MODE <= 1;
1679 end
1680
1681
1682
1683
1684
1685 endmodule // cmd_ctl
1686

```